International Journal of Electronics,
Communication & Instrumentation Engineering
Research and Development (IJECIERD)
ISSN(P): 2249-684X; ISSN(E): 2249-7951
Vol. 3, Issue 5, 2013, Dec 2013, 1-10
© TJPRC Pvt. Ltd.

A NOVEL HIGH-SPEED, HIGHER-ORDER 128 BIT ADDERS FOR DIGITAL SIGNAL PROCESSING APPLICATIONS USING ADVANCED EDA TOOLS

K. SRAVYA[1] & K. MURTHY RAJU[2]

[1]Research Scholar, VLSID, Shri Vishnu Engineering College for Women, Bhimavaram,
West Godavari District, Andhra Pradesh, India

[2]Department of Electronics & Communication, Shri Vishnu Engineering College for Women, Bhimavaram,
West Godavari District, Andhra Pradesh, India

ABSTRACT

In digital adders, the speed of addition is limited by the time required to transmit a carry through the adder. Carry Select Adder (CSLA) is one of the fastest adders used in many data-processing processors to perform fast arithmetic functions. Now days we use the very High Speed IC Designs In that designs we required very High Speed Structures. Parallel prefix adder is the most flexible and widely used for binary addition. Parallel Prefix adders are best suited for VLSI implementation. Numbers of parallel prefix adder structures have been proposed over the past years intended to optimize area, fan-out, logic depth and inter connect count. This paper presents a new approach to redesign the basic operators used in parallel prefix architectures. Based on this modification 8, 16, 32, 64 and 128-bit Kogge-Stone Parallel Prefix Adders architectures have been developed and compared with the regular and Modified CSLA architecture. The modified CSLA design has reduced area and Power as compared with the regular CSLA but delay is increased. The Proposed Kogge-Stone adder has less delay when compared with regular and modified CSLA. This work estimates the performance of the proposed designs in terms of delay, area are implemented in Xilinx ISE.

KEYWORDS: High Speed VLSI, CSLA, BEC, Parallel Prefix Adder, Dot Operators' Parallel Prefix Adder

INTRODUCTION

The main area of research in VLSI system design are Area and power reduction in data path logic systems. The main fundamental requirement of high-performance processors and systems is High-speed addition and multiplication. Due to the time required to propagate a carry through the adder, the speed of addition is limited in digital adders. In an elementary adder the sum for each bit position is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position. The production of carries is the major speed limitation in any adder.

The Carry Select Adder is used in many computational systems to moderate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum. However, the Carry Select Adder is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input and then the final sum and carry are selected by the multiplexers (mux). To overcome the above problem, n-bit binary to excess-1 code converter (BEC) is used to improve the speed of addition. This logic can be implemented with any type of adder to further improve the speed.. The main advantage of this BEC logic comes from the lesser number of logic gates than the Full Adder (FA) structure.

In electronics, The carry-select adder is one of the simple but rather faster, having a gate level depth of $O(\sqrt{n})$ in computing $(n+1)$-bit sum of two n-bit numbers.

The regular carry-select adder generally consists of two ripple carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with the two ripple carry adders. One ripple carry adder is used to perform the calculation with the assumption of the carry being zero and the other ripple carry adder is used to perform the calculation assuming carry one. After the two results are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known. The number of bits in each carry select block can be uniform, or variable.

Basic Building Block

The basic building block of a carry-select adder is as shown in Figure 1 where the block size is 4. Two 4-bit ripple carry adders are multiplexed together, where the resulting carry and sum bits are selected by the value of carry-in. One ripple carry adder sum is considered when carry-in is 0, and the other sum is considered when carry-in is 1.

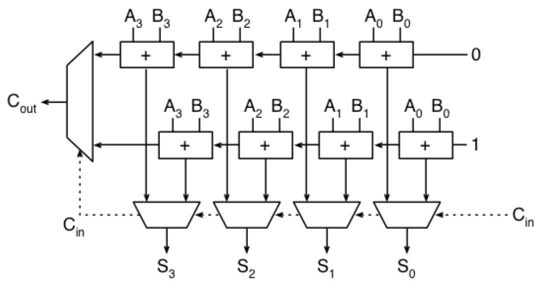


Figure 1: Basic Carry Select Adder

Uniform-Sized Adder

A 16-bit carry-select adder with a uniform block size of 4 can be created with three of these blocks and a 4-bit ripple carry adder. Since carry-in is known at the beginning of computation, a carry select block is not needed for the first four bits. The delay of this adder will be four full adder delays, plus three MUX delays.

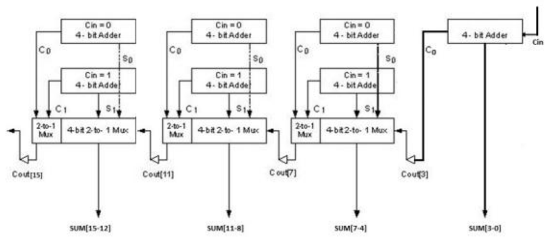


Figure 2: Regular Fixed Size CSLA

Variable-Sized Adder

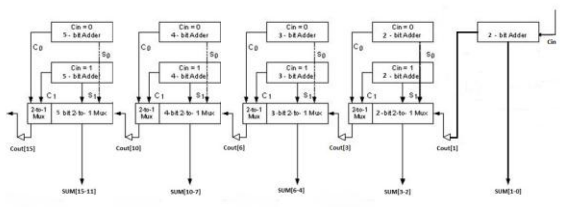


Figure 3: Variable Sized CSLA

The above shown figure is a 16-bit carry-select adder with variable size with block sizes of 2-2-3-4-5. This break-up is ideal when the full-adder delay is equal to the MUX delay, which is unlikely. The total delay is two full adder delays, and four mux delays.

BASIC STRUCTURE OF REGULAR 16-BIT CSLA

A 16-bit carry select has two types of block size namely uniform block size and variable block size. A 16-bit carry select adder with a uniform block size has the delay of four full adder delays and three MUX delays. While a 16-bit carry select adder with variable block size has the delay of two full adder delays, and four mux delays. Therefore we use 16-bit carry select adder with variable block size. Ripple-carry adders are the simplest and most compact full adders, but their performance is limited by a carry that must ripple from the least-significant to the most-significant bit. A carry-select adder achieves speeds 40% to 90% faster by performing additions in parallel and reducing the maximum carry path.

A carry-select adder is divided into sectors, each of which, except for the least significant performs two additions in parallel, one assuming a carry-in of zero, the other a carry-in of one within the sector, there are two 4-bit ripple- carry adders receiving the same data inputs but different Cin. The upper adder has a carry-in of zero, the lower adder a carry-in of one. The actual Cin from the preceding sector selects one of the two adders. If the carry-in is zero, the sum and carry-out of the upper adder are selected. If the carry-in is one, the sum and carry-out of the lower adder are selected. Logically, the result is not different if a single ripple-carry adder were used.

First the coding for full adder and different multiplexers of 6:3, 8:4, 10:5, and 12:6 was done. Then 2, 3, 4, 5-bit ripple carry adder was done by calling the full adder. The regular 128-bit CSLA was created by calling the ripple carry adders and all multiplexers based on circuit.

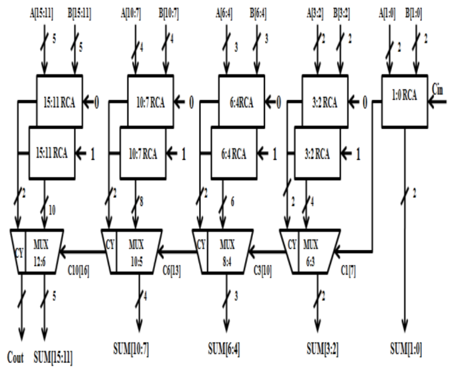


Figure 4: Regular 16-bit SQRT CSLA

BASIC STRUCTURE OF MODIFIED 16-BIT CSLA

It is similar to regular 16-bit SQRT CSLA. Only change is that in basic blocks having two ripple-carry adders, one ripple carry adder fed with a constant 1 carry-in is replaced by BEC. The area estimation of each group is calculated.

Based on the consideration of delay values, the arrival time of selection input C1 [time (T) =7] of 6:3 mux is earlier than the s3 [t =9] and c3 [t =7] and later than the s2 [t =4]. Thus, the sum3 and final c3 (output from mux) are depending on s3 and mux and partial c3 (input to mux) and mux, respectively. The sum2 depends on c1 and mux. For the remaining parts the arrival time of mux selection input is always greater than the arrival time of data inputs from the BEC's. Thus, the delay of the remaining MUX depends on the arrival time of mux selection input and the mux delay.

First the coding for full adder and multiplexers of 6:3, 8:4, 10:5, and 12:6 was done. The BEC program was design by using NOT, XOR and AND gates. Then 2, 3, 4, 5-bit ripple carry adder was done by calling the full adder. The modified 16-bit CSLA was created by calling the ripple carry adders, BEC and all multiplexers based upon the circuit. Finally, modified 128-bit CSLA was created by calling the ripple carry adders, BEC and all multiplexers based on circuit.

The arrival time of selection input of 6:3 mux is earlier. Thus, the sum3 and final c3 (output from mux) depends on s3 and mux and partial c3 (input to mux) and mux, respectively. The sum2 depends on c1and mux. For the remaining group's the arrival time of mux selection input is always greater than the arrival time of data inputs from the BEC's. Thus, the delay of the remaining groups depends on the arrival time of mux selection input and the mux delay.

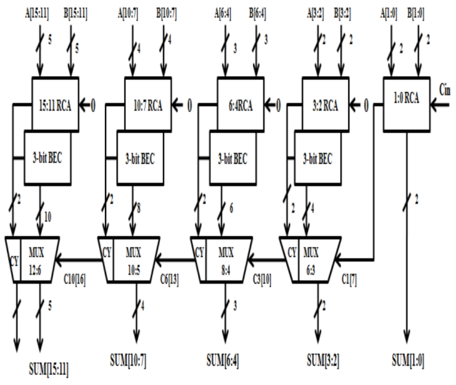


Figure 5: Modified 16-bit SQRT CSLA

PROPOSED METHOD

The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor data path units. As such, extensive research continues to be focused on improving the power delay performance of the adder. In VLSI implementations, parallel-prefix adders are known to have the best performance. Reconfigurable logic such as Field Programmable Gate Arrays (FPGAs) has been gaining in popularity in recent years because it offers improved performance in terms of speed and power over DSP-based and microprocessor-based solutions for many practical designs involving mobile DSP and telecommunications applications and a significant reduction in development time and cost over Application Specific Integrated Circuit (ASIC) designs. The power advantage is especially important with the growing popularity of mobile and portable electronics, which make extensive use of DSP functions. However, because of the structure of the configurable logic and routing resources in FPGAs, parallel-prefix adders will have a different performance. In particular, most modern FPGAs employ a fast-carry chain which optimizes the carry path for the simple Ripple Carry Adder (RCA).

In this paper, the practical issues involved in designing and implementing tree-based adders on FPGA's. An efficient testing strategy for evaluating the performance of these adders is discussed.

KOGGE-STONE ADDER

The arrangement of the prefix network specifies the type of the Parallel Prefix Adder.The Kogge-Stone tree achieves both $\log_2 N$ stages and fan-out of 2 at each stage. This comes at the cost of long wires that must be routed between stages. The tree also contains more PG cells; while this may not impact the area if the adder layout is on a regular grid, it will increase power consumption. Despite these cost, Kogge-Stone adder is generally used for wide adders because it shows the lowest delay among other structures.

The complete functioning of Kogge Stone Adder can be easily comprehended by analyzing it in terms of three distinct parts

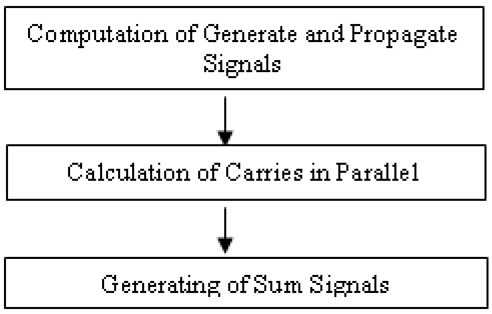

Computation of Generate and Propagate Signals
Calculation of Carries in Parallel
Generating of Sum Signals

Figure 6: Steps in Parallel Prefix Adder

Pre Processing

This step involves computation of generate and propagate signals corresponding too each pair of bits in A and B. These signals are given by the logic equations below:

$\mathbf{p_i = A_i \text{ xor } B_i}$

$\mathbf{g_i = A_i \text{ and } B_i}$

Carry Look Ahead Network

This block differentiates Kogge Stone Adder from other adders and is the main force behind its high performance. This step involves computation of carries corresponding to each bit. It uses group propagate and generate as intermediate signals which are given by the logic equations below:

$\mathbf{P_{i:j} = P_{i:k+1} \text{ and } P_{k:j}}$

$\mathbf{G_{i:j} = G_{i:k+1} \text{ or } (P_{i:k+1} \text{ and } G_{k:j})}$

Post Processing

This is the final step and is common to all adders of this family (carry look ahead). It involves computation of sum bits. Sum bits are computed by the logic given below:

`$\mathbf{S_i = p_i \text{ xor } C_{i-1}}$

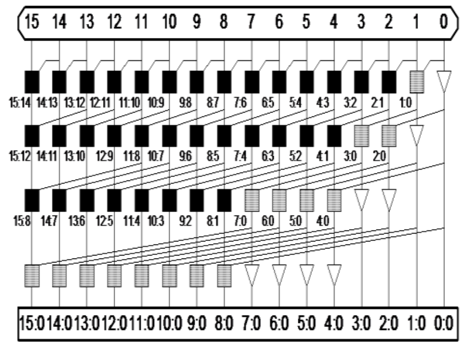


Figure 7: 16 Bit Kogge-Stone Parallel Prefix Adder

IMPLEMENTATION RESULTS

The design proposed in this paper has been developed using Verilog-HDL, Simulation Done by using Model Sim 6.5 e and synthesized in Xilinx ISE 9.1i.

The Regular CSLA, Modified CSLA and Kogge Stone Adders Simulation and Synthesis Wave forms show in below figures. The 128 bit Modified Carry Select adder shown figure 10 and 13 .It compare to regular CSLA modified CSLA architecture is take less amount of delay it is approximately reduced by 30% of regular method design, But if we go for the Proposed Design (Kogge-Stone Adder) Delay is Less Than 50% of Modified CSLA, Less Than 65% of Regular CSLA and nearly 85% less Delay as Compare Ripple Carry Adder, The Delay Results and Graphs are shown in Table 1 and Figure 14

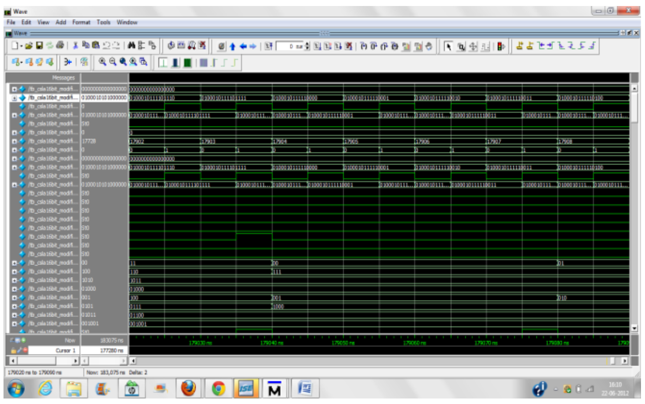


Figure 8: Regular 128-Bit CSLA

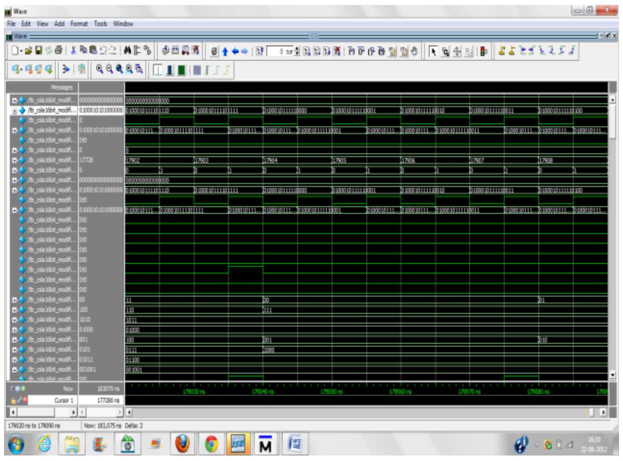


Figure 9: Modified 128-Bit CSLA

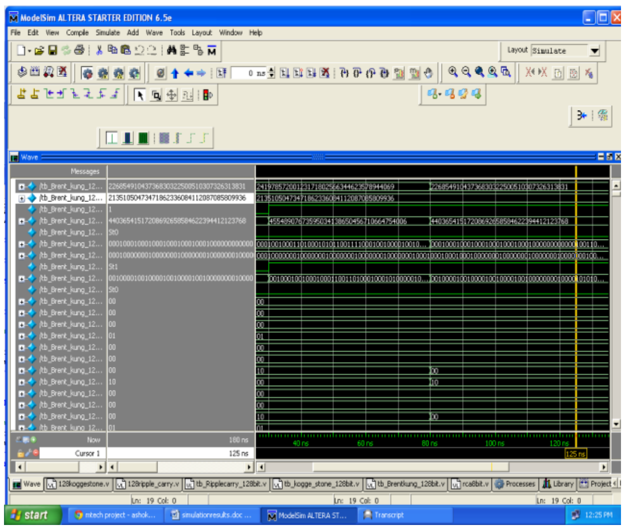
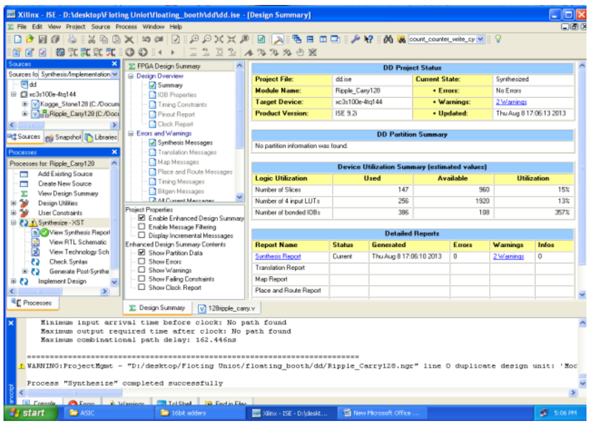


Figure 10: 128 Bit Kogge - Stone Adder


Xilinx - ISE - D:\desktop\Floting Uniot\floating_booth\dd\dd.ise - [Design Summary]

DD Project Status			
Project File:	dd.ise	**Current State:**	Synthesized
Module Name:	Ripple_Carry128	**• Errors:**	No Errors
Target Device:	xc3s100e-4tq144	**• Warnings:**	2 Warnings
Product Version:	ISE 9.2i	**• Updated:**	Thu Aug 8 17:06:13 2013

DD Partition Summary
No partition information was found.

Logic Utilization	Used	Available	Utilization
Number of Slices	147	960	15%
Number of 4 input LUTs	256	1920	13%
Number of bonded IOBs	386	108	357%

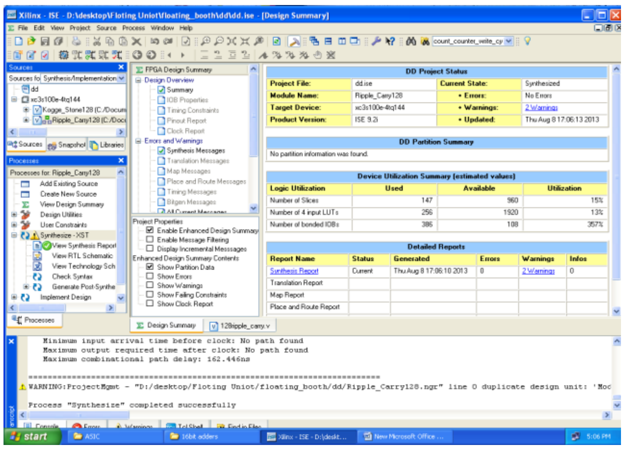
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Thu Aug 8 17:06:10 2013	0	2 Warnings	0
Translation Report					
Map Report					
Place and Route Report					

```
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 162.446ns

=========================================================================
WARNING:ProjectMgmt - "D:/desktop/Floting Uniot/floating_booth/dd/Ripple_Carry128.ngr" line 0 duplicate design unit: 'Moc

Process "Synthesize" completed successfully
```

Figure 11: 128 Bit Regular CSLA


Xilinx - ISE - D:\desktop\Floting Uniot\floating_booth\dd\dd.ise - [Design Summary]

DD Project Status			
Project File:	dd.ise	**Current State:**	Synthesized
Module Name:	Ripple_Carry128	**• Errors:**	No Errors
Target Device:	xc3s100e-4tq144	**• Warnings:**	2 Warnings
Product Version:	ISE 9.2i	**• Updated:**	Thu Aug 8 17:06:13 2013

DD Partition Summary
No partition information was found.

Logic Utilization	Used	Available	Utilization
Number of Slices	147	960	15%
Number of 4 input LUTs	256	1920	13%
Number of bonded IOBs	386	108	357%

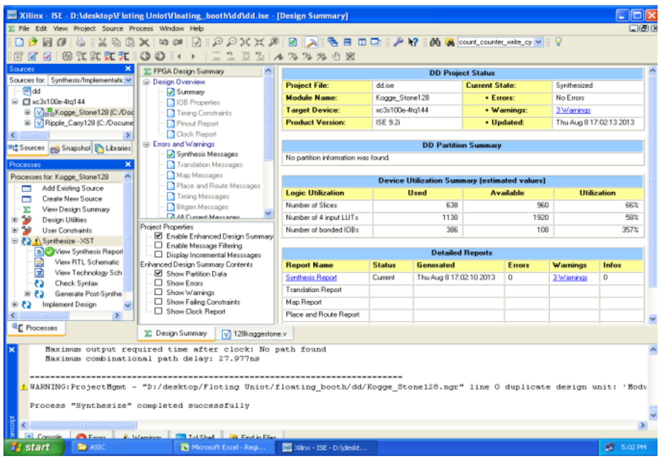
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Thu Aug 8 17:06:10 2013	0	2 Warnings	0
Translation Report					
Map Report					
Place and Route Report					

```
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 162.446ns

=========================================================================
WARNING:ProjectMgmt - "D:/desktop/Floting Uniot/floating_booth/dd/Ripple_Carry128.ngr" line 0 duplicate design unit: 'Moc

Process "Synthesize" completed successfully
```

Figure 12: 128 Bit Modified CSLA


Xilinx - ISE - D:\desktop\Floting Uniot\floating_booth\dd\dd.ise - [Design Summary]

DD Project Status			
Project File:	dd.ise	**Current State:**	Synthesized
Module Name:	Kogge_Stone128	**• Errors:**	No Errors
Target Device:	xc3s100e-4tq144	**• Warnings:**	3 Warnings
Product Version:	ISE 9.2i	**• Updated:**	Thu Aug 8 17:02:13 2013

DD Partition Summary
No partition information was found.

Logic Utilization	Used	Available	Utilization
Number of Slices	638	960	66%
Number of 4 input LUTs	1130	1920	58%
Number of bonded IOBs	386	108	357%

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Thu Aug 8 17:02:10 2013	0	3 Warnings	0
Translation Report					
Map Report					
Place and Route Report					

```
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 27.977ns

=========================================================================
WARNING:ProjectMgmt - "D:/desktop/Floting Uniot/floating_booth/dd/Kogge_Stone128.ngr" line 0 duplicate design unit: 'Modu

Process "Synthesize" completed successfully
```

Figure 13: 128 Bit Kogge-Stone Adder

Table 1: Comparisons of Adders

Delay (Ns)	RCA	CSLA Regular	CSLA Modified	Kogge Stone
16 Bit	23.540	18.420	16.562	16.481
32 Bit	44.366	25.536	21.411	21.027
64 Bit	83.726	39.588	25.950	27.167
128 Bit	162.446	56.375	40.562	27.977

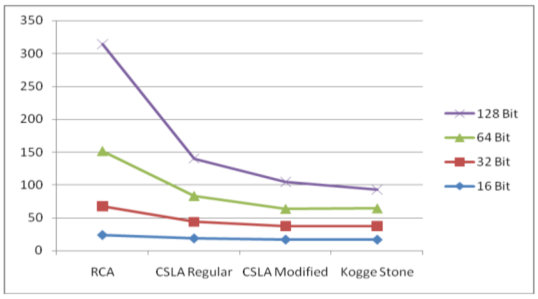


Figure 14: Comparing Graph for All Adders

CONCLUSIONS

A simple modification is maid to regular CSLA to reduce the Delay and area of CSLA architecture. The reduced number of gates of this work offers the great advantage in the reduction of area and also the total power. The compared results show that the modified CSLA has less delay and it requires less area consumption, but this CSLA not suitable for Digital Signal Processing Applications because DSP applications require Very Fast computations design. Therefore Kogge-Stone Parallel Prefix Adder is proposed, it has nearly High Speed Computation as Compared to Modified CSLA. The speed of the proposed adder is almost double the existing adders.

In This Paper all the Adders RCA, CSLA Regular, CSLA Modified, Kogge-Stone with 16 Bit, 32 Bit, 64 Bit and 128 Bit Sizes are developed.

With Respect of Results and Tables Finally it can be concluded that For Higher Adder Applications Proposed Adder is Best Choice because it has 500 % high Speed compare CSLA Designs. As the functional verification decides the quality of the silicon, we spend 60% of the design cycle time only for the verification/simulation. In order to avoid the delay and meet the TTM, we use the latest verification methodologies.

REFERENCES

1. Bedrij, O. J., (1962), “Carry-select adder,” IRE Trans. Electron. Comput., pp.340–344 .
2. Ceiang ,T. Y. and Hsiao,M. J. , “Carry-select adder using single ripple carry adder,” Electron. Lett., vol. 34, no. 22, pp. 2101– 2103
3. Ramkumar, B., Kittur, H.M. and Kannan, P. M. ,(2010),“ASIC implementation of modified faster carry save adder,” Eur. J. Sci. Res., vol. 42, no. 1,pp.53–58.
4. Kogge P, Stone H, “A parallel algorithm for the efficient solution of a general class Recurrence relations”, IEEE Trans. Computers, vol.C-22, No.8, pp 786-793, Aug.1973.
5. P. M. Kogge and H. S. Stone, “A parallel algorithm for the efficient solution of a general class of recurrence equations,” IEEE Trans. Computers, vol. 22, no. 8, pp. 786–793, August 1973.
6. J. Liu, S. Zhou, H. Zhu, and C.-K. Cheng, “An algorithmic approach for generic parallel adders,” in ICCAD, November 2003, pp. 734–730.
7. R. Zimmermann, “Non-heuristic optimization and synthesis of parallel-prefix adders,” in International Workshop on Logic and Architecture Synthesis, December 1996, pp. 123–132.
8. T. Matsunaga and Y. Matsunaga, “Timing-constrained area minimization algorithm for parallel prefix adders,” IEICE Transactions on Fundamentals, vol. E90-A, no. 12, pp. 2770–2777, December 2007.

9. T. Matsunaga, S. Kimura, and Y. Matsunaga, “Power-conscious syntheses of parallel prefix adders under bitwise timing constraints,” in Proc. the Workshop on Synthesis And System Integration of Mixed Information technologies(SASIMI), Sapporo, Japan, October 2007, pp. 7–14.

10. Kim ,Y. and Kim ,L.-S.,(May2001), “64-bit carry-select adder with reduced area,” Electron Lett., vol. 37, no. 10, pp. 614–615.

11. E. Abu-Shama and M. Bayoumi, “A New cell for low power adders,” in Proc.Int. Midwest Symp. Circuits and Systems, 1995, pp. 1014–1017

12. Verilog HDL- Digital Design and Synthesis, by Samith Palnitkar